APPLICATION OF AN AUTOMATIC IMPLEMENTATION METHOD OF LOTOS SPECIFICATION TO TRANSPORT LAYER PROTOCOL

4L−3

Pairoj TERMSINSUWAN , Zi-xue CHENG , Norio SHIRATORI
TOHOKU University

1.INTRODUCTION

There are many problems in implementation of LOTOS specification because LOTOS is a very high abstract language. Some of them have already been discussed in ref.[1] and [2] . In this paper , we are going to discuss the problems which are left and in order to solve them, we would like to propose a supporting system for helping the implementer to create the remaining program fragments necessary in implementation.

2.OUTLINE OF THE RESEARCH

2.1 Previous researches

In implementation of LOTOS specification , there are several problems for implementer to overcome as shown in Fig.1. First, we try to derive program automaically from LOTOS specification as much as possible. Ref[1] provides an automatic translation algorithm from process part of LOTOS specification to C program. This derived program has some parts depending on OS functions. In ref[2] ,we have discussed an idea to introduce an OS interface between the program and OS so that the program is now independent of any specific OS functions.

2.2 Unsolved problems

Any way, only the program concerning to process part is not enough , the other two parts which are necessary for implementation are

[A] ADT part This is program fragment for abstract data type(ADT) part of LOTOS specification.

[B] Additional Information This is program fragment for additional part not written in LOTOS. (for instance: detail of internal action)

Up to now, these two parts have been written manually depending on application and OS case

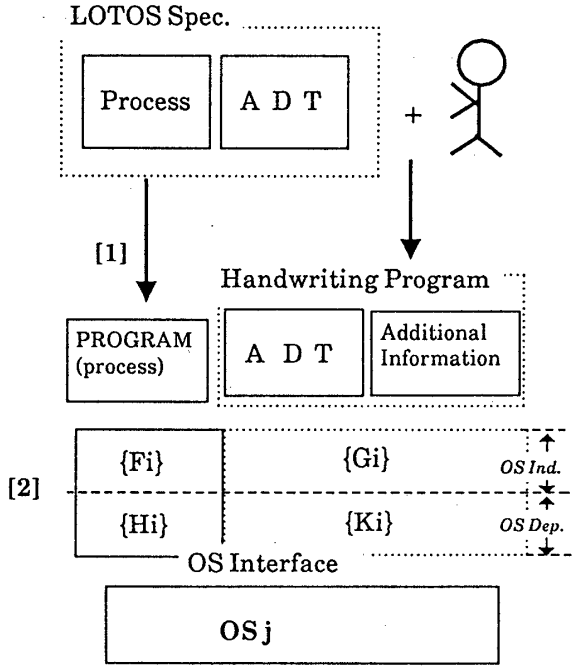

Fig.1 Outline of our research

by case, then implementers have to write these parts for every specification and for every OS.

2.3 Our proposal

We propose an interface between these two parts and OS, and a supporting system for creating the interface as well as programs. This interface will become the execution environment for programs corresponding to LOTOS specification and make programs of the process, ADT and additional information part independent of OS. By interactively communicating with this supporting system, we can semiautomatically obtain the program according to the process ,ADT, additional information .

3.DESIGN OF THE SUPPORTING SYSTEM

Now, we try to design a supporting system for helping the implementer to create these three programs and OS interface. The supporting system

has the structure consisting of four main parts as shown in Fig.2.

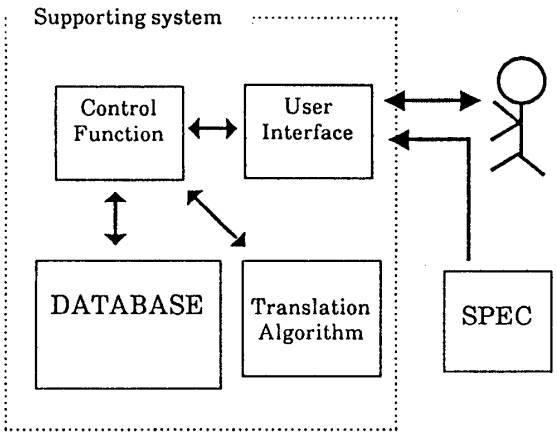

Fig. 2 Structure of the supporting system

[1] Translation algorithm. This is the translation algorithm from process part of LOTOS specification to C program mentioned in (2.1) . LOTOS specification will be translated automatically.

[2] Database. The primitive functions needed for creating the ADT and additional part are contained in this database, for example, function of how to implement natural number used in LOTOS ,etc. An implementer will select some functions depending on the case and modify them so that a set of modified functions becomes the desirable program.

[3] User interface. We provide a user-friendly interface with a user based on multi-window environment using templates.

[4] Control function. This part will control the other three parts to create the program according to the specification.

4. DESIGN EXAMPLE FOR TRANSPORT LAYER PROTOCOL

In this example, we are going to design the supporting system, especially for the database, by taking transport layer protocol as a model for study. First, the process part will be translated to C program by translation algorithm, then the implementer has to create the programs according to ADT and additional information parts by selecting some default functions in database and modifying them.

Regarding to ADT part of the protocol, we provide two groups of primitive functions in the database which are functions concerning to

[1] Data types in standard library of LOTOS (for example, Boolean, Set,String,etc.) and

[2] Data types existing in general OSI model. Some data types of basic entity in OSI model are ①Adderss (in this case, Transport Address) ② Service Primitive (like Connection Request, Connection indication) and ③Protocol Data Unit or PDU (Connection Request PDU, etc.)

We show two examples of common primitive functions used in these data types;

(1) **make**(*Ename, E1,...,En*)

This is a function for making new entity(*Ename*) from other entities (*E1* to *En*),for example,making Trasport address (*T__address__sort*) from Network address prefix (*N__address__sort*) and Transport service access point indicating suffix (*T__suffix__sort*) written in the protocol as

opns *make__T__addr : N__address__sort, T__suffix__sort → T__address__sort*

(2) **project**(*i*, **make**(*Ename, E1,...,En*))

This is a function for drawing the *i*th entity (*Ei*, $1 \leqq i \leqq n$) from entity *Ename*, for example , drawing Source address (*T__address__sort*) from Transport connection indication (*TCONind* consisting of *T__address__sort* and request option(*options__sort*)) written in the protocol as

opns *make__TCONind : T__address__sort, options__sort → TCONind*

source__address : TCONind → T__address__sort

There still may be the part which can not be created by the primitive functions. In this case, the implementer has to create and add it to the process part by handwriting.

5. CONCLUSION

In this paper, we have discussed the outline of the supporting system.The detailed design of this model is now under developing.

References:

[1]Zixue CHENG, Kaoru TAKAHASHI,Norio SHIRATORI, Shoichi NOGUCHI, "An Automatic Implementation Method of Protocol Specification in LOTOS,"Technical Report,IN90-49.

[2]Pairoj TERMSINSUWAN , Zi-xue CHENG , Norio SHIRATORI, "Implementation of LOTOS specification and improvement of run time support routine"情報処理学会第43回全国大会 講演論文集(1), pp.191 - 192 (1991)